MixDQ: Memory-Efficient Few-Step Text-to-Image Diffusion Models with Metric-Decoupled Mixed Precision Quantization

Tianchen Zhao[12*], Xuefei Ning[1*†], Tongcheng Fang[12*], Enshu Liu[1], Guyue Huang[3], Zinan Lin[4], Shengen Yan[2], Guohao Dai[25], and Yu Wang[1†]

[1] Tsinghua University
[2] Infinigence AI
[3] University of California Santa Barbara
[4] Microsoft Research
[5] Shanghai Jiaotong University
{suozhang1998, foxdoraame, sgcbflrftc, hguyue1, linzinan1995}@gmail.com, les19@mails.tsinghua.edu.cn, yanshengen@infini-ai.com, daiguohao@sjtu.edu.cn, yu-wang@mail.tsinghua.edu.cn

Abstract. Few-step diffusion models, which enable high-quality text-to-image generation with only a few denoising steps, have substantially reduced inference time. However, considerable memory consumption (5-10GB) still poses limitations for practical deployment on mobile devices. Post-Training Quantization (PTQ) proves to be an effective method for enhancing efficiency in both memory and operational complexity. However, when applied to few-step diffusion models, existing methods designed for multi-step diffusion face challenges in preserving both visual quality and text alignment. In this paper, we discover that the quantization is bottlenecked by highly sensitive layers. Consequently, we introduce a mixed-precision quantization method: **MixDQ**. Firstly, we identify some highly sensitive layers are caused by outliers in text embeddings, and design a specialized Begin-Of-Sentence (BOS)-aware quantization to address this issue. Subsequently, we investigate the drawback of existing sensitivity metrics, and introduce metric-decoupled sensitivity analysis to accurately estimate sensitivity for both image quality and content. Finally, we develop an integer-programming-based method to obtain the optimal mixed-precision configuration. In the challenging 1-step Stable Diffusion XL text-to-image task, current quantization methods fall short at W8A8. Remarkably, MixDQ achieves **W3.66A16** and **W4A8** quantization with negligible degradation in both visual quality and text alignment. Compared with FP16, it achieves **3-4**× reduction in model size and memory costs, along with a **1.5**× latency speedup. The project URL is https://a-suozhang.xyz/mixdq.github.io/ .

Keywords: Diffusion Model · Quantization

[*] Equal contribution
[†] Corresponding Authors: Yu Wang and Xuefei Ning

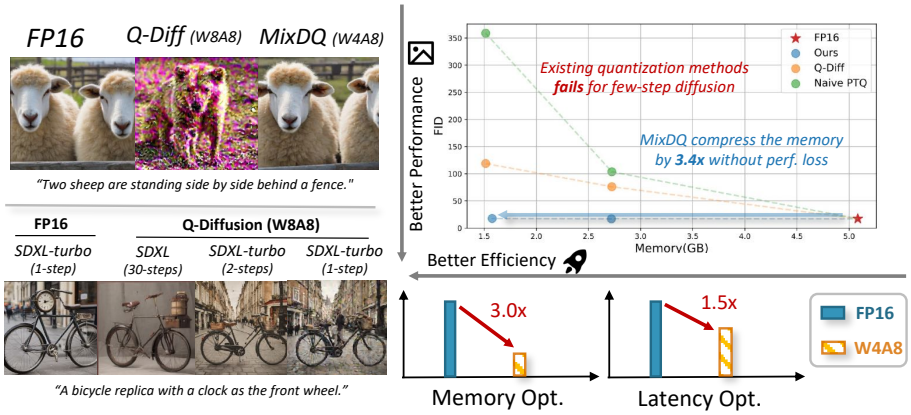


Fig. 1: The effectiveness of MixDQ. Left: MixDQ preserves both image quality and image-text alignment. Right: The efficiency improvements of MixDQ.

1 Introduction

Text-to-image diffusion models [23, 26, 27] attract substantial attention for their ability to generate high-quality images from textual prompts. However, their high computational and memory demands present challenges for real-time application and deployment on mobile devices [1, 48]. Recent efforts on few-step diffusion models [15, 31, 35] have significantly alleviated the computational burden, which require only 1-8 steps to generate high-fidelity images compared to the 10-100 steps in previous approaches [7, 34]. However, their memory cost remains excessive [28, 39]. For example, running a Stable Diffusion XL-turbo model [23] in the 16-bit floating-point (FP16) format with a batch size of 1 consumes a peak memory of 9.7GB, which exceeds the capacity of many mobile devices and even some desktop GPUs (e.g., an RTX 4070 has 8GB GPU memory).

Model quantization [9], compressing high bit-width floating-point parameters and activations into lower bit-width integers, proves to be an effective strategy for reducing both memory and computational cost. Many prior researches [3, 4] explore quantization for diffusion models. However, we observe that **few-step diffusion models are more sensitive to quantization than multi-step ones**. As depicted in Fig. 1, the quantized SDXL 30-step model and SDXL-turbo 2-step model have notably less degradation compared with the SDXL-turbo 1-step model. Prior research Q-diffusion [11] performs well on multi-step models but encounters challenges with W8A8 quantization for one-step SDXL-turbo [31]. The resulting image appears blurred and contains numerous artifacts. We conjecture this might be due to the absence of the iterative denoising process, which could compensate for the quantization error.

Additionally, prior research primarily focuses on preserving the image quality for quantized models. However, as illustrated in Fig. 1 and Fig. 2, quantization impacts not only image quality but also content. The altered content may lead to

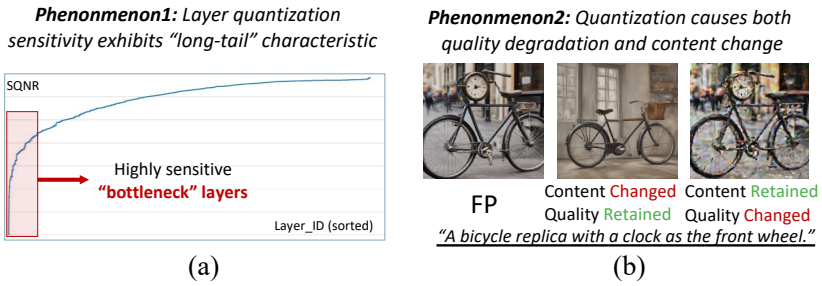


Fig. 2: Insightful findings for few-step text-to-image diffusion quantization. Left: The layer-sensitivity distribution has a "long-tail" characteristic. Right: Quantization affects both image quality and content.

degradation in image-text alignment, which refers to how well the generated image aligns with the given text instruction. For instance, in Fig. 1, the image generated by Q-diffusion depicts a polar bear, which contradicts the prompt mentioning "goats". Similarly, the middle image in Fig. 2 loses the content "clock".

To effectively quantize the few-step model while preserving the quality and alignment in the meantime, we investigate the reasons behind the failure of existing quantization methods. We discover two insightful phenomena, as presented in Fig. 2. Firstly, we measure each layer's quantization sensitivity for the SDXL-turbo model with the signal-to-quantization-noise ratio (SQNR) following prior research [46]. As shown in Fig. 2 (a), the distribution exhibits a "long-tail" characteristic. Consequently, using uniform quantization settings for all layers, as in previous research, would result in the quantization being "bottlenecked" by some highly sensitive layers. We look into these sensitive layers and propose: (1) designing a specialized quantization method to "protect" the outliers in certain layers, significantly reducing their quantization error; and (2) adopting mixed-precision quantization to "protect" the sensitive layers with a higher bit-width, thus achieving higher quality. Secondly, as illustrated in Fig. 2 (b), quantization affects both the image content and quality. We identify that the entanglement of these two factors in evaluation will cause quantization methods to fail in preserving generation quality. Therefore, we propose "decoupling" the effects of quality and content in the sensitivity evaluation. This will be elaborated in Sec. 3.2.

In light of above, we introduce a mixed-precision quantization method: **MixDQ**. Firstly, we analyze the characteristics of highly sensitive layers and discover that many of them are associated with the quantization of text embeddings. We further investigate the feature distribution of text embeddings and design a specialized **Begin-of-sentence (BOS)-aware quantization** (Sec. 3.1) to address the outlier values. Secondly, we identify the shortcomings of existing quantization sensitivity evaluation and design an improved **Metric-decoupled sensitivity analysis** (Sec. 3.2) based on the idea of decoupling the impact of quantization on image content and quality. Finally, we design an **integer-programming-**

based mixed precision allocation (Sec. 3.3) to acquire optimal bit-width configuration based on the improved sensitivity analysis.

We summarize the contributions of this paper as follows:

1. We highlight the challenge of quantizing the few-step diffusion model, compared to quantizing the multi-step diffusion model. Additionally, we emphasize the often-overlooked necessity of preserving alignment when compressing the text-to-image generative models.
2. Based on the careful investigation of the data distribution and sensitivity of each layer, we design MixDQ, a mixed precision quantization method with improved sensitivity evaluation and quantization techniques.
3. We evaluate MixDQ in two settings: weight-only quantization and normal weight and activation quantization. MixDQ can achieve W4A16 and W5A8 quantization with a negligible 0.1 FID increase. Compared with the FP16 model, MixDQ can achieve W3.66A16 and W4A8 quantization within a 0.5 FID increase, resulting in a 3× memory cost reduction and a 1.5× latency speedup on Nvidia GPUs.

The techniques in MixDQ could benefit future research and applications of compression methods on other generative models and tasks. Firstly, BOS-aware quantization addresses the issue of outlier values in text embeddings, which is an inherent problem for transformer-based models (also recognized as the "attention sink" [44] in language models). Secondly, we believe the explicit and decoupled consideration of various metrics is important whenever compressing visual generative models. On one hand, from an application perspective, multiple metrics should be considered, especially for generative tasks. On the other hand, for better compression results, the explicit and decoupled consideration of multiple metrics can avoid the failure of compression methods.

2 Related Work

Diffusion Models. [7, 33, 36] could generate high-quality images through an iterative denoising process. However, the excessive cost of repeated denoising iterations calls for improvements in the sampling efficiency, namely, reducing the timesteps. Some research [14, 34, 47] focuses on designing improved numerical solvers, and another line of research explores utilizing distillation [15, 17, 30, 31] to condense the sampling trajectory into fewer steps or even a single step. These few-step diffusion models significantly reduce the computational cost, however, as discussed in Sec. 1, **they face additional challenges for quantization**.

Network Quantization. Prior research, such as PTQD [4] and Q-DM [12] explores utilizing quantization techniques on diffusion models. Q-Diffusion [11] extends this application to large-scale stable diffusion models. Other research [8, 32, 37] continues to improve post-training quantization techniques on aspects like calibration and temporal adjustment. However, the majority of existing diffusion quantization methods **employs uniform bit-with for layers with diverse**

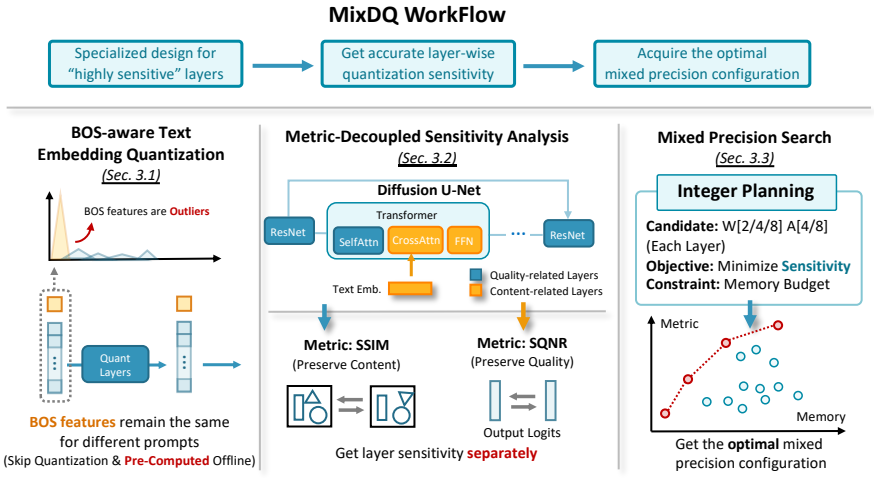


Fig. 3: Framework of the proposed mixed-precision quantization method: MixDQ. It consists of three key components, the BOS-aware quantization addresses the highly sensitive text embedding, the metric-decoupled scheme improves sensitivity analysis, and the integer programming acquires the optimal bit-width allocation.

sensitivity. Yang et. al. [46] suggest preserving sensitive layers as high-precision based on SQNR metric. However, we observe that the SQNR metric tends to prioritize content change, potentially leading to quality degradation (as discussed in Sec. 3.2). Inspired by the prior success of mixed precision [2, 40], we design a mixed precision quantization method to address the imbalanced sensitivity.

Evaluation Metrics. The currently widely used metrics for evaluating text-to-image generation can be summarized into two major aspects: image fidelity (quality) and image-text alignment. To estimate image fidelity, metrics such as FID [6] and IS [29] are commonly used. These metrics measure the feature space distance between generated images and reference images. For image-text alignment, CLIP Score [5] is often utilized to calculate the similarity of the image and the text embeddings. To align the statistical scores and human preference, ImageReward [45] and HPS [43] collect user preference and train the model to predict generated image scoring, considering both fidelity and alignment. While previous methods solely discuss preserving the image quality, we investigate quantization's effect on **both image fidelity and alignment**.

3 Methods

Fig. 3 presents our mixed precision quantization method **MixDQ** consisting of three consecutive steps. Firstly, we identify the highly sensitive layers and examine their distinctive properties. Based on these properties, we devise specialized

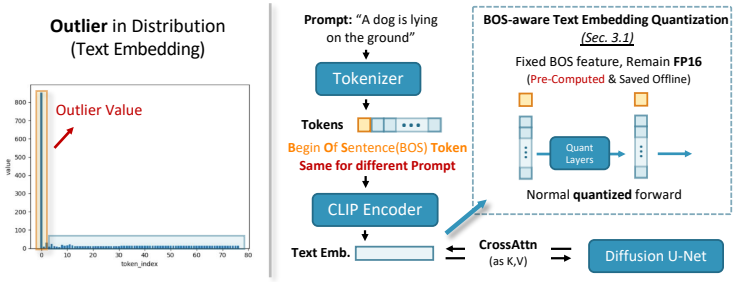


Fig. 4: Illustration of BOS-aware Quantization. Left: the first token has a significantly larger value than the others. Right: Since BOS token features remain the same for different prompts, we skip quantizing them and pre-compute them offline.

quantization techniques tailored for them (Sec. 3.1). Secondly, in Sec. 3.2, we introduce a metric-decoupled analysis that measures the quantization's effects on image content and quality separately. Finally, based on the sensitivity, we employ integer programming to determine the optimal mixed precision configuration under a given budget (Sec. 3.3).

3.1 BOS-aware Text Embedding Quantization

Fig. 2 shows that the diffusion model quantization is "bottlenecked" by some highly sensitive layers. Upon going through the highly sensitive layers depicted in the "long-tail", we find that a substantial portion of them corresponds to the "to_k" and "to_v" linear layers in the cross-attention. All of these layers take the text embedding, which is the output of the CLIP encoder [25], as their input. Fig. 4 (Left) shows the maximum magnitude of the text embedding of each token (average across 8 sentences). We observe that the 1st token has a significantly larger magnitude (823.5) compared to the rest of the tokens (10-15). Quantizing this tensor with such an outlier value to 8-bit would lead to the majority of values being close to 0, resulting in the loss of crucial textual information. This observation could potentially explain why current quantization methods struggle to maintain text-image alignment in Fig. 1 (Left).

As shown in Fig. 4 (Right), the first outlier token in the CLIP output corresponds to the "Begin Of Sentence (BOS)" in the tokenizer output. Actually, *the feature of this BOS token remains the same across different prompts*. Therefore, we can skip the quantization and computation for the "to_k/v" layers for this token. Concretely, we pre-compute the floating-point output feature of the "to_k/v" layers for the BOS token and then concatenate it with the dequantized features of other tokens. In this way, the quantization error of the CLIP embedding is significantly reduced as the outlier is removed. Additionally, only 640-1280 (the channel number of BOS features) elements need to be stored for each "to_k/v" layer, introducing only negligible overhead.

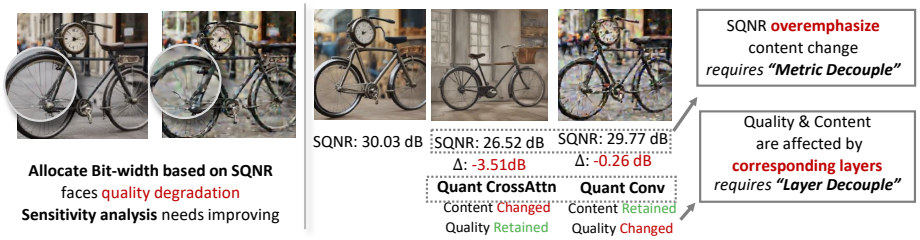


Fig. 5: Decoupling metrics and layers to separate the influence on image quality and content. Left: Existing SQNR-based sensitivity analysis needs improving. Right: SQNR's problem of overemphasizing content change.

3.2 Metric-Decoupled Sensitivity Analysis

Preliminary Experiment and Analysis. We start by designing preliminary experiments. As discussed in Sec. 1, we measure the sensitivity with SQNR. It is computed as the ratio of the L2 norm of feature values to the quantization noise. The quantization noise is estimated by evaluating the impact on the network's output logits with each layer quantized. We observe the "long-tailed" characteristics of sensitivity distribution in Fig. 2 (a). An intuitive solution is to allocate higher bit-widths for highly sensitive layers and lower bit-widths for less sensitive ones. We implement a straightforward bit-width allocation method based on this principle to achieve an average of W8A8. However, as seen in Fig. 5 (Left), the generated image faces severe quality degradation, suggesting that the current sensitivity evaluation relying solely on SQNR may require refinement.

We look into this issue and observe the following phenomena in Fig. 5 (Right): (1) Both changes in image content and quality impact the SQNR. In particular, compared with the 30.03dB image, the SQNR decline of an image with changed contents but high quality (-3.51dB) is much more significant than that of an image with similar contents but unacceptable quality (-0.26dB). This means that *when evaluating images with varying content and quality, SQNR tends to underestimate the performance degradation resulting from decreased image quality.* (2) *Image quality and content are mainly affected by different layer types.* Specifically, quantizing the cross-attention layers causes content changes, but the quality remains good. This aligns with the intuition that text supervision predominantly interacts with cross-attention layers, which control the image content. Conversely, when quantizing convolutions, the quality degrades and the content is preserved. Further details of the correlation between layer types and their impact on content or quality are provided in the supplementary.

As a result, during the bit-width allocation process, since SQNR tends to overemphasize content changes, the content-related layers are retained at very high bit-widths. Consequently, the bit-width for quality-related layers is decreased. This *"unfair competition" between quality- and content-related layers* explains the severe degradation of the image quality observed in Fig. 5 (Left) when the bit-width allocation is solely based on SQNR.

Method Design. Inspired by these findings, we improve the sensitivity analysis by decoupling metrics and layers to separate the quantization's effect on quality and content. Therefore, we separate the layers into two groups: the content-related layers, i.e., the cross-attention layers and feed forward networks (FFNs), and the quality-related layers, i.e., self-attention layers and convolutions. Then, we conduct sensitivity analysis for each group separately with distinct metrics.

When performing the sensitivity analysis for the content-related layers, we adopt the Structural Similarity Index Measure (SSIM) [42] to assess image content change. Given a generated image x and reference image y, The SSIM metric combines the three components, luminance (l), contrast (c) and structure (s):

$$\begin{aligned} l(x,y) = \frac{2\mu_x\mu_y}{\mu_x^2+\mu_y^2}, c(x,y) = \frac{2\sigma_x\sigma_y}{\sigma_x^2+\sigma_y^2}, s(x,y) = \frac{\sigma_{xy}}{\sigma_x\sigma_y}, \\ \text{SSIM}(x,y) = l(x,y)^\alpha \cdot c(x,y)^\beta \cdot s(x,y)^\gamma, \end{aligned} \tag{1}$$

where μ, σ represent the mean and variance of pixel values; and α, β, γ are weights (set to 1) that control three components. We use the full image size as the window size for SSIM. Unlike Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) that measure absolute errors, SSIM is adept at perceiving changes in structural information, making it well-suited for measuring content change.

When conducting sensitivity analysis for the quality-related layers, we utilize the SQNR metric. Given that quantizing only the quality-related layers does not significantly alter the image content, SQNR serves as a suitable metric for assessing the degree of image quality degradation in this scenario.

3.3 Integer Programming Bit-width Allocation

Having obtained the layer sensitivity, we can determine the mixed precision configuration by assigning higher bit-widths to more sensitive layers. Enlightened by prior research [10], we formulate the bit-width allocation into an integer programming problem: Given the resource budget $\mathcal{B}$, and candidate bit-widths $b \in \{2,4,8\}$, we aim to determine the bit-width choices $\boldsymbol{c}$ (a one-hot indicator for each layer) that maximizes the sum of sensitivity $\mathcal{S} = \sum_{i=1}^{N} \sum_{b=2,4,8} c_{i,b} \cdot \mathcal{S}_{i,b}$:

$$\begin{aligned} &\underset{c_{i,b}}{\text{argmax}} \quad \sum_{i=1}^{N} \sum_{b=2,4,8} c_{i,b} \cdot \mathcal{S}_{i,b} \\ &\text{s.t.} \quad \sum_{b=2,4,8} c_{i,b} = 1, \quad \sum_{i=1}^{N} \sum_{b=2,4,8} c_{i,b} \cdot \mathcal{M}_{i,b} \leq \mathcal{B}, \\ &\quad c_{i,b} \in \{0,1\}, \quad \forall i \in \{1,\cdots,N\}, \forall b \in \{2,4,8\}, \end{aligned} \tag{2}$$

where N is the number of layers in the model; $c_{i,b} = 1$ indicates that the i-th layer will be quantized to b-bit, and $\mathcal{S}_{i,b}$ is the corresponding sensitivity score (higher SQNR or SSIM is better). $\mathcal{M}_{i,b}$ denotes the resource cost of the i-th layer when it is quantized to b-bit.

We conduct the above integer programming separately for activation quantization and weight quantization, as well as for each of the two layer groups, i.e., the content-related cross-attention and FFN layers, and the quality-related self-attention and convolution layers.

4 Experiments

4.1 Experimental Settings

Quantization Scheme. We adopt the simplest and easily deployable asymmetric min-max quantization scheme similar to [19,22]. The quantization parameters (i.e., scaling factor, zero point) are shared within each tensor for the activation or within each output channel for the weight. The shortcut-splitting quantization technique [11] is applied to all methods in Tab. 1. We randomly sampled 1024 prompts from COCO [13] as our calibration dataset.

Mixed Precision Allocation. We conduct metric-decoupled sensitivity analysis by measuring the quantization effect on output sensitivity metrics (SQNR and SSIM) when quantizing certain layers and preserving the remaining layers as FP. The sensitivity is averaged over 32 randomly sampled prompts. the search can be completed within seconds. We use the averaged bit-width weighted by layer param size as the budget for integer programming and adopt 2, 4, 8 as candidate bit-width choices. We do not quantize all nonlinear activation and normalization layers, and quantize all the linear and convolution layers. In our metric-decoupled sensitivity analysis, we iterate through all layers one by one. For each layer, our method quantizes only that layer and preserves the remaining layers as FP. Then, we measure two types of sensitivity metrics for this layer: SQNR and SSIM. The sensitivity score is averaged over 32 prompts. We quantize each layer to three bit-width choices: 2, 4, 8, and get their corresponding sensitivity scores. Then, we set a budget for the average bit-width of all elements (weights or activation) and use the layer-wise sensitivity scores to set up the integer programming. We use the OR-tools [38] library for integer programming. The efficiency of this implementation allows the bit-width allocation in seconds.

Hardware Profiling. We measure the latency and memory usage of MixDQ on the Nvidia RTX 4080 GPU using CUDA 12.1. All profiling is conducted with a batch size of 1. Specifically, we measure memory usage for all models using the PyTorch Memory Management APIs [24]. The inference latency of the FP16 models is profiled with NVIDIA Nsight tools [20]. For quantized models, we measure the latency of the quantized layers, the quantization operation, and the unquantized layers separately, and add them up to calculate the overall latency. We profile the FP and quantized models using stable-fast [16], a toolkit that provides state-of-the-art inference speed for diffuser models. We use the kernels from the Cutlass [21] library to implement the quantized layers and develop a quantization GPU kernel to reduce the quantization overhead.

Table 1: Performance and efficiency comparison of MixDQ and other quantization methods on full COCO annotations. The "CLIP" and "IR" denotes CLIP Score and ImageReward metric. The "Storage Opt." and "Compute Opt." denote equivalent savings of model size and computational complexity (measured in Bit Operations as in [22]). The bit-width "16" represents FP16 without quantization. The "weight only" setting represents the rows with activation bit-width of FP16, and the rest are the "normal" weight-activation quantization.

Model	Method	Bit-width (W/A)	*Storage Opt.*	*Compute Opt.*	FID(↓)	CLIP(↑)	IR(↑)
SDXL-turbo (1 step)	FP	16/16	-	-	17.15	0.2722	0.8631
	Naive PTQ	8/16	2×	1×	16.89	0.2740	0.8550
		4/16	4×	1×	301.49	0.1581	-2.2526
		8/8	2×	4×	103.96	0.1478	-1.7446
		4/8	4×	8×	358.894	0.1242	-2.2815
	Q-Diffusion	8/16	2×	1×	16.97	0.2735	0.8588
		4/16	4×	1×	22.58	0.2685	0.6847
		8/8	2×	4×	76.18	0.1772	-1.3112
		4/8	4×	8×	118.93	0.1662	-1.6353
	MixDQ(Ours)	4/16	4×	1×	17.23	0.2693	0.8254
		3.66/16	4.4×	1×	17.40	0.2682	0.7528
		8/8	2×	4×	17.03	0.2703	0.8415
		5/8	3.2×	8×	17.23	0.2697	0.8307
		4/8	4×	8×	17.68	0.2698	0.7822
LCM-lora (4 steps)	FP	16/16	-	-	25.56	0.2570	0.2122
	Naive PTQ	8/8	2×	4×	23.36	0.2548	0.0517
		4/8	4×	8×	87.36	0.2055	-1.6160
	Q-Diffusion	8/8	2×	4×	23.92	0.2561	0.1875
		4/8	4×	8×	57.73	0.2280	-1.1863
	MixDQ(Ours)	8/8	2×	4×	22.54	0.2552	0.1573
		4/8	4×	8×	33.48	0.2403	-0.6732

4.2 Performance and Efficiency Comparison

We conduct experiments on widely-used few-step diffusion models, SDXL-turbo, and LCM-Lora, for text-to-image generation tasks using COCO2014 at the resolution of 512×512. We adopt three different metrics: FID for fidelity, CLIP Score for image-text alignment, and ImageReward for human preference. The metrics are calculated on all 40,504 prompts. For the baseline methods, "naive PTQ" and "Q-diffusion," we use a uniform bit-width for all layers. For MixDQ, we calculated the average bit-width weighted by each layer's parameter size. Following [22], we measure the theoretical computational savings ("Compute Opt.") in Bit Operations (BOPs). The "Storage Opt." represents the model size reduction. The comparison of the performance and resource consumption of MixDQ quantization is presented in Tab. 1 and Fig. 6.

We experiment with two quantization settings: the "weight-only" scheme which seeks larger compression rate of model size, and the "normal" scheme focuses on both storage saving and latency speedup. During activation quanti-

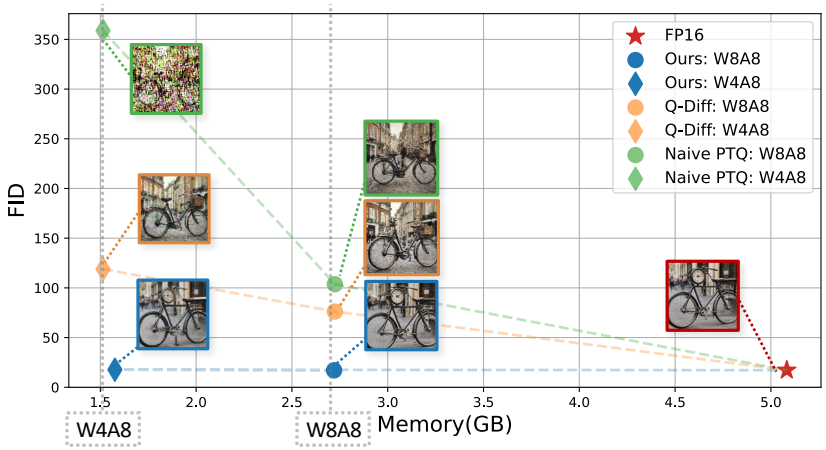


Fig. 6: The FID with respect to memory cost of MixDQ and baseline quantization methods, with corresponding generated images. MixDQ achieves lossless quantization, whereas baseline methods fail to generate readable images.

zation, we observed that certain layers retain sensitivity and cannot be quantized to 8 bits without sacrificing performance. To address this, we choose to retain 1% of the most sensitive layers based on metric-decoupled sensitivity. Fig. 7 shows that it introduces minimal overhead but ensures the preservation of performance.

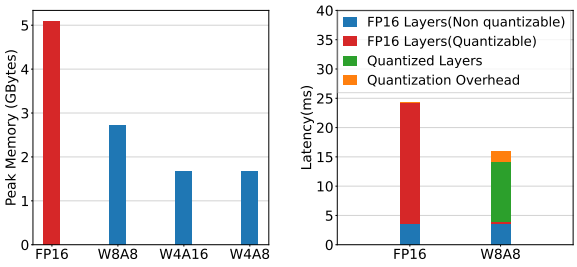
As evident from Fig. 6 and Tab.1, for the SDXL-turbo model, the baseline quantization methods can only maintain image quality with W8A16. The naive PTQ's FID increases drastically from around 17.15 to 103.96 and 301.49 for W8A8 and W4A16, respectively. Q-diffusion, with the assistance of Adaround [18], manages to preserve performance for W4A16 but still fails at W8A8, generating images with "oil-painting"-like quality degradation. For W4A8, both PTQ and Q-diffusion produce images that are hardly readable, exhibiting FID values exceeding 100, negative ImageReward, and CLIP Score below 0.2. In contrast, MixDQ generates images that are nearly identical to FP16 images, maintaining both the image content and quality. Even with W4A8 quantization, MixDQ incurs a 0.5 FID increase and a 2.5e-3 CLIP Score drop. As for LCM-Lora, the baseline methods encounter fewer failures since they involve 4 iteration steps. Nevertheless, MixDQ consistently outperforms them for all metrics.

We conduct further experiments to compare the performance of MixDQ with other quantization techniques under W8A8. The results are provided in Section 1 of the supplementary material.

4.3 Hardware Resource Savings

Memory footprint reduction. Fig. 7 (a) shows the GPU memory usage of MixDQ and the FP16 baseline on UNet inference. MixDQ can reduce the memory footprint from two aspects. Firstly, quantizing the model weights leads to smaller allocated memory to store the UNet model. Secondly, quantizing the ac-

Bit-width (W/A)	Memory Opt.	Latency Opt.	FID (↓)
16/16	-	-	17.15
4/16	3.03×	-	17.23
8/8	1.87×	1.45×	17.03
4/8	3.03×	1.45×	17.68

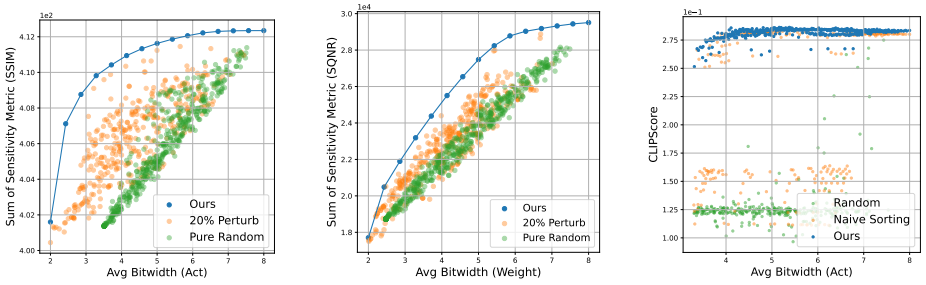


(a) (b)

Fig. 7: The illustration of MixDQ's hardware resource savings. Left: The comparison of efficiency and performance under different MixDQ mixed precision configurations. Right: (a) MixDQ's optimization of peak memory, (b) MixDQ's latency breakdown under W8A8.

tivations saves allocated memory to store residual connection activations. Combining the two benefits, we can effectively reduce the peak memory footprint by 1.87×, 3.03× and 3.03× under the W8A8, W4A16 and W4A8, respectively.

Speedup of model inference. Fig. 7 (a) shows the latency of the UNet model inference on Nvidia RTX 4080. We use the W8A8 scenario to showcase the speedup. MixDQ W8A8 can accelerate the inference of UNet by 1.52× over the FP16 baseline. 7 (b) gives a breakdown of the inference latency. There are three types of layers in the model: quantizable layers, including linear and convolution layers; non-quantizable layers, such as normalization and non-linear activation layers; and quantization layers, that perform the FP16-to-int8 conversion. The non-quantizable layers have the same latency in both the baseline and MixDQ. The quantizable layers are accelerated by 1.97×, approximately the same ratio between INT8 and FP16 hardware peak throughput on RTX4080 (2×). MixDQ requires quantization layers for converting activations, but even with this overhead, the end-to-end acceleration still reaches 1.52×.



(a) (b) (c)

Fig. 8: The Pareto frontier of mixed precision configurations. The x-axis represents the averaged bit-width, the y-axis of Subfigures (a), (b), (c) presents the sensitivity metrics (SQNR & SSIM), and the final evaluation metric (CLIPScore).

5 Analysis

5.1 Analysis of Paretor Frontier

The Pareto frontier [41] is often applied to present the accuracy-efficiency trade-off. We present the scatter plot of three metrics with respect to averaged bit-width for SDXL-turbo in Fig. 8. In Fig. 8 (a),(b), we examine the effectiveness of integer planning by choosing the sensitivity metric (SQNR and SSIM) as the y-axis. As can be seen, compared with the "Pure random" baseline that randomly chooses bit-width, the "Ours" configuration achieves a significantly better trade-off. Furthermore, we randomly "perturb" 20% of layers' bit-width in our configuration by replacing it with another bit-width and observing lower metric scores, denoting the superiority of our bit-width allocation. Furthermore, to demonstrate that our integer programming is non-trivial, we design a "Naive Sorting" baseline that repeatedly lowers the bit-width of the least sensitive layer to fit the average bit-width. We compare the final evaluation metric (CLIP Score) of "Ours" result, "Naive Sorting," and "Random" in Fig. 8 (c). As can be seen, our result achieves a superior trade-off, demonstrating not only the effectiveness of the integer programming but also the accuracy of our acquired sensitivity.

5.2 Ablation Studies

We conduct ablation studies by gradually incorporating MixDQ techniques into the W8A8 quantized SDXL-turbo model. As illustrated in Fig. 9 and Tab.2, the generated images improve from severe degradation to surpassing FP ones. The Pareto frontier in Fig. 8 could also assist in proving the effectiveness. We discuss the effectiveness of the proposed techniques individually as follows:

Effectiveness of BOS-aware Quantization. When adopting naive W8A8 PTQ for SDXL-turbo, as shown in Fig. 9 and Tab. 2, the generated quality degrades, and the content changes significantly. After introducing BOS-aware quantization, the image content is recovered, and the metric values improve significantly (FID: 103.95 $\rightarrow$ 31.65, CLIP Score: 0.1478 $\rightarrow$ 0.2652).

Effectiveness of Metric-Decoupled Sensitivity. In Fig. 9 (c), we present the result of the bit-width allocation based on SQNR without our metric-decoupled scheme. It can be observed that compared with Fig. 9 (b), the quality worsened (FID: 31.65 $\rightarrow$ 37.35, CLIP Score: 0.2652 $\rightarrow$ 0.2624) after applying mixed precision. This reveals the insufficient accurate sensitivity and highlights the importance of our metric-decoupled technique. Also, Fig. 8 (c) helps prove the metric-decoupled sensitivity has a high correlation with final evaluation metrics.

Effectiveness of Mixed Precision. As shown in Tab.2 and Fig. 9 (d), when applying mixed precision with our metric-decoupled sensitivity analysis, MixDQ achieves lossless quantization with generated images nearly identical to FP and acquires similar metric values (FID: 17.03 vs. 17.15, CLIP Score: 0.2703 vs. 0.2722). Fig. 8 (a),(b) also illustrates that our integer programming strikes the optimal performance-efficiency trade-off on the Pareto frontier.

Table 2: Ablation studies on MixDQ techniques. By gradually incorporating the proposed techniques on SDXL-turbo with W8A8 quantization, the generated images exhibit improvements from failure to surpassing the FP16 baseline.

BOS-aware	*Mixed-Precision*	*Metric-Decouple*	Bit-width(W/A)	FID (↓)	CLIP (↑)
-	-	-	FP16	17.15	0.2722
-	-	-	8/8	103.96	0.1478
✓	-	-	8/8	31.65	0.2652
✓	✓	-	8/8	37.35	0.2624
✓	✓	✓	8/8	17.03	0.2703

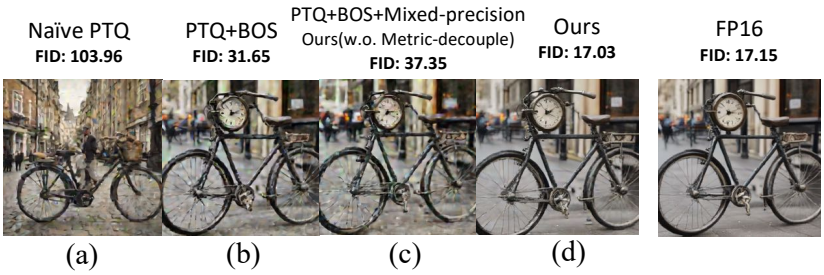


Fig. 9: The illustration of ablation studies on MixDQ techniques. From left to right, as techniques are progressively added, notable improvement in both generated image quality and alignment is witnessed.

6 Conclusions

We propose a mixed-precision quantization method: MixDQ, which consists of three steps. Firstly, BOS-aware text embedding quantization addresses the highly sensitive layers. Secondly, metric-decoupled sensitivity analysis is introduced to consider preserving both the image quality and content. Finally, an integer programming is conducted for bit-width allocation. MixDQ achieves W4A8 with negligible performance loss, and practical hardware speedup.

Acknowledgements

This work was supported by National Natural Science Foundation of China (No. 62325405, 62104128, U19B2019, U21B2031, 61832007, 62204164), Tsinghua EE Xilinx AI Research Fund, and Beijing National Research Center for Information Science and Technology (BNRist). We thank for all the support from Infinigence-AI.

References

1. Chen, Y.H., Sarokin, R., Lee, J., Tang, J., Chang, C.L., Kulik, A., Grundmann, M.: Speed is all you need: On-device acceleration of large diffusion models via gpu-aware optimizations. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) pp. 4651–4655 (2023), https://api.semanticscholar.org/CorpusID:258298971
2. Dong, Z., Yao, Z., Gholami, A., Mahoney, M.W., Keutzer, K.: Hawq: Hessian aware quantization of neural networks with mixed-precision. 2019 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 293–302 (2019), https://api.semanticscholar.org/CorpusID:148571720
3. He, Y., Liu, J., Wu, W., Zhou, H., Zhuang, B.: Efficientdm: Efficient quantization-aware fine-tuning of low-bit diffusion models. ArXiv **abs/2310.03270** (2023), https://api.semanticscholar.org/CorpusID:263672060
4. He, Y., Liu, L., Liu, J., Wu, W., Zhou, H., Zhuang, B.: Ptqd: Accurate post-training quantization for diffusion models. ArXiv **abs/2305.10657** (2023), https://api.semanticscholar.org/CorpusID:258762678
5. Hessel, J., Holtzman, A., Forbes, M., Bras, R.L., Choi, Y.: Clipscore: A reference-free evaluation metric for image captioning. arXiv preprint arXiv:2104.08718 (2021)
6. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium **30** (2017)
7. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems **33**, 6840–6851 (2020)
8. Huang, Y., Gong, R., Liu, J., Chen, T., Liu, X.: Tfmq-dm: Temporal feature maintenance quantization for diffusion models. ArXiv **abs/2311.16503** (2023), https://api.semanticscholar.org/CorpusID:265466808
9. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A.G., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 2704–2713 (2017), https://api.semanticscholar.org/CorpusID:39867659
10. Li, S., Ning, X., Hong, K., Liu, T., Wang, L., Li, X., Zhong, K., Dai, G., Yang, H., Wang, Y.: LLM-MQ: Mixed-precision Quantization for Efficient LLM Deployment. In: NeurIPS 2023 Efficient Natural Language and Speech Processing Workshop (2023)
11. Li, X., Lian, L., Liu, Y., Yang, H., Dong, Z., Kang, D., Zhang, S., Keutzer, K.: Q-diffusion: Quantizing diffusion models. ICCV (2023)
12. Li, Y., Xu, S., Cao, X., Sun, X., Zhang, B.: Q-dm: An efficient low-bit quantized diffusion model. In: Neural Information Processing Systems (2023), https://api.semanticscholar.org/CorpusID:268096292
13. Lin, T.Y., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014), https://api.semanticscholar.org/CorpusID:14113767
14. Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J.: Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. arXiv preprint arXiv:2206.00927 (2022)
15. Luo, S., Tan, Y., Patil, S., Gu, D., von Platen, P., Passos, A., Huang, L., Li, J., Zhao, H.: Lcm-lora: A universal stable-diffusion acceleration module. ArXiv **abs/2311.05556** (2023), https://api.semanticscholar.org/CorpusID:265067414

16. fast maintainers, S.: Stable-fast, https://github.com/chengzeyi/stable-fast
17. Meng, C., Rombach, R., Gao, R., Kingma, D., Ermon, S., Ho, J., Salimans, T.: On distillation of guided diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14297–14306 (2023)
18. Nagel, M., Amjad, R.A., van Baalen, M., Louizos, C., Blankevoort, T.: Up or down? adaptive rounding for post-training quantization. ArXiv **abs/2004.10568** (2020), https://api.semanticscholar.org/CorpusID:216056295
19. Nagel, M., Fournarakis, M., Amjad, R.A., Bondarenko, Y., van Baalen, M., Blankevoort, T.: A white paper on neural network quantization. ArXiv **abs/2106.08295** (2021), https://api.semanticscholar.org/CorpusID:235435934
20. NVIDIA: Nsight Systems. https://docs.nvidia.com/nsight-systems/index.html, https://docs.nvidia.com/nsight-systems/index.html
21. Nvidia: Nvidia cutlass release v3.4 (2024), https://github.com/NVIDIA/cutlass
22. Pandey, N.P., Nagel, M., van Baalen, M., Huang, Y.R., Patel, C., Blankevoort, T.: A practical mixed precision algorithm for post-training quantization. ArXiv **abs/2302.05397** (2023), https://api.semanticscholar.org/CorpusID:256808627
23. Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Muller, J., Penna, J., Rombach, R.: Sdxl: Improving latent diffusion models for high-resolution image synthesis. ArXiv **abs/2307.01952** (2023), https://api.semanticscholar.org/CorpusID:259341735
24. PyTorch: PyTorch Memory Management (2023), https://pytorch.org/docs/stable/notes/cuda.html#memory-management
25. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
26. Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-shot text-to-image generation. In: ICML. pp. 8821–8831. PMLR (2021)
27. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR. pp. 10684–10695 (2022)
28. Ryu, H., Lim, S., Shim, H.: Memory-efficient personalization using quantized diffusion model. ArXiv **abs/2401.04339** (2024), https://api.semanticscholar.org/CorpusID:266899926
29. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans **29** (2016)
30. Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. arXiv preprint arXiv:2202.00512 (2022)
31. Sauer, A., Lorenz, D., Blattmann, A., Rombach, R.: Adversarial diffusion distillation. ArXiv **abs/2311.17042** (2023), https://api.semanticscholar.org/CorpusID:265466173
32. So, J., Lee, J., Ahn, D., Kim, H., Park, E.: Temporal dynamic quantization for diffusion models. ArXiv **abs/2306.02316** (2023), https://api.semanticscholar.org/CorpusID:259075274
33. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International Conference on Machine Learning. pp. 2256–2265. PMLR (2015)
34. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502 (2020)

35. Song, Y., Dhariwal, P., Chen, M., Sutskever, I.: Consistency models. In: International Conference on Machine Learning (2023), https://api.semanticscholar.org/CorpusID:257280191
36. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456 (2020)
37. Tang, S., Wang, X., Chen, H., Guan, C., Wu, Z., Tang, Y., Zhu, W.: Post-training quantization with progressive calibration and activation relaxing for text-to-image diffusion models. ArXiv **abs/2311.06322** (2023), https://api.semanticscholar.org/CorpusID:265149927
38. Team, G.O.T.D.: Or-tools. https://github.com/google/or-tools (Year Accessed)
39. Wang, H., Shang, Y., Yuan, Z., Wu, J., Yan, Y.: Quest: Low-bit diffusion model quantization via efficient selective finetuning. ArXiv **abs/2402.03666** (2024), https://api.semanticscholar.org/CorpusID:267500241
40. Wang, K., Liu, Z., Lin, Y., Lin, J., Han, S.: Haq: Hardware-aware automated quantization with mixed precision. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 8604–8612 (2018), https://api.semanticscholar.org/CorpusID:102350477
41. Wikipedia: Pareto front (2024), https://en.wikipedia.org/wiki/Pareto_front, accessed: March 3, 2024
42. Wikipedia contributors: Structural similarity index measure. https://en.wikipedia.org/wiki/Structural_similarity_index_measure, accessed on: February 28, 2024
43. Wu, X., Sun, K., Zhu, F., Zhao, R., Li, H.: Better aligning text-to-image models with human preference. arXiv preprint arXiv:2303.14420 (2023)
44. Xiao, G., Tian, Y., Chen, B., Han, S., Lewis, M.: Efficient streaming language models with attention sinks. ArXiv **abs/2309.17453** (2023), https://api.semanticscholar.org/CorpusID:263310483
45. Xu, J., Liu, X., Wu, Y., Tong, Y., Li, Q., Ding, M., Tang, J., Dong, Y.: Imagereward: Learning and evaluating human preferences for text-to-image generation. arXiv preprint arXiv:2304.05977 (2023)
46. Yang, Y., Dai, X., Wang, J., Zhang, P., Zhang, H.: Efficient quantization strategies for latent diffusion models. ArXiv **abs/2312.05431** (2023), https://api.semanticscholar.org/CorpusID:266163100
47. Zhao, W., Bai, L., Rao, Y., Zhou, J., Lu, J.: Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. arXiv preprint arXiv:2302.04867 (2023)
48. Zhao, Y., Xu, Y., Xiao, Z., Hou, T.: Mobilediffusion: Subsecond text-to-image generation on mobile devices. ArXiv **abs/2311.16567** (2023), https://api.semanticscholar.org/CorpusID:265466277